A Software Framework to Enhance Training and Operations of Space Missions

Bradley J. Betts[1], Richard Papasin[2], Sharif Elcott[3], Dawn McIntosh[2], Rommel del Mundo[3], Brian Niehaus[3], Robert W. Mah[2], Michael Guerrero[4], Edward Wilson[5]

Smart Systems Research Laboratory
NASA Ames Research Center, M/S 269-1
Moffett Field, CA 94035-1000

[1]Computer Sciences Corporation, bbetts@email.arc.nasa.gov
[2]NASA, {Richard.I.Papasin, Robert.W.Mah, Dawn.M.McIntosh}@nasa.gov
[3]QSS Group, Inc., {sharif, rdelmundo, bniehaus}@email.arc.nasa.gov
[4]Guerrero Engineering, mguerrer@email.arc.nasa.gov
[5]Intellization, Ed.Wilson@Intellization.com

Abstract

Many software development tasks centered on training and operations for spaceflight have common aspects. Software engineers and researchers working in these areas stand to benefit from a software framework that can provide frequently used components. Preliminary results from the development of such a framework are presented in this paper. It is hoped that by documenting design patterns and providing reusable software components, cost savings can be realized and design experience can be captured. Components in the framework are grouped into four major areas: Information Management, Visualization, Simulation and Decision Making, and Real-time Data. An example is presented of the framework being applied to address remote training challenges at the NASA Space Station Training Facility.

1. Introduction

An important part of current and future space missions is increasing the ease and reducing the cost of training and operations. For example, astronauts should be able to participate in useful training while at remote locations. That is, a particular crew may have members on assignment at the NASA Johnson Space Center in Houston and at Star City, Russia. Without having to physically bring the crew together, they should be able to engage in meaningful collaborative training sessions. In the operational realm, ground controllers should be able to easily visualize the current, future, and past configurations of a space platform and be able to rapidly gain complete information about onboard systems.

Delivering these kinds of capabilities at a reasonable cost is a challenge. However, the risk reductions and cost savings derived from more efficient training and operations will help offset the development and maintenance outlays for hardware and software systems. Software costs in particular can be vexing—software developed to meet training or operational requirement may be difficult to reuse to solve a new challenge. The software profession as a whole has recognized the problem of efficient code reuse and has responded with research and development into various toolkits, frameworks, and design patterns [1-4].

This paper describes preliminary results for a software framework and prototype application being developed at the NASA Ames Research Center by the Smart Systems Research Laboratory. The goal of the effort, referred to as the Intelligent Virtual Station (IVS), is to provide a software framework flexible and powerful enough to support the training and operational needs of a wide variety of space missions. A *software framework* is defined to be a reusable collection of classes and designs cooperating to meet the challenges of an application area. The principal users of the IVS framework would be software developers and researchers working in the area of spaceflight. It will be used to develop applications of direct use to those involved in operations and training.

The work presented here has thus far been restricted to dealing with training and operational problems related to the International Space Station (ISS) [5]. The ISS is a large and complex international effort that poses a variety of information technology challenges. Prototype application development targeted at the ISS is being done to exercise, validate, and provide feedback to the design

Back to TOC

of the framework before broadening it to include other space missions such as the Space Shuttle Program, Strategic Launch Initiative, and other future space platforms.

2. Framework Overview

Several information technology disciplines are involved in advancing the state-of-the-art in training and operations. These include areas such as visualization, information management, simulation and decision-making tools, and real-time data. For many training and operations scenarios, individual users will continue to make use of these IT disciplines via conventional computer interfaces: a monitor, keyboard, mouse, and client-side processing capability. Other interfaces, such as head-mounted devices and projection displays, are seeing increased use, although their cost and relative lack of portability decrease their usefulness in some situations (e.g., remote training).

Delivering these IT disciplines to users poses a challenge for software engineers and researchers. Since training and operations applications have many aspects in common, reinventing capabilities would waste time and money. This is particularly true of software used for spaceflight projects. Components must be rigorously designed and exhaustively tested; those that present a UI must adhere to set styles. All these factors contribute to higher costs.

As an example, a software engineer might be faced with designing the client-side components of an operations application to be used by ground controllers to monitor the performance of a particular spacecraft. To meet the requirements, the engineer would like to present users with a detailed virtual environment (VE) of the spacecraft. The VE will be used by ground controllers to monitor the real-time state of a critical set of onboard systems, and users will interact with the application via the conventional computer interfaces previously described. The engineer looks to reuse the VE and real-time data access components to meet the requirements. A second engineer, designing a second operations application, may hope to reuse only the real-time data access component.

Both engineers would likely be aided by an environment as outlined in the UML [6, 7] diagram shown in Figure 1. Client-side components encompassing the desired features are available for use. These components might be available in different formats (e.g., JAR files or .NET assemblies) and include documentation of their public interfaces, design, and any design patterns they employ. The client-side components in turn interface with server-side resources (e.g., databases, real-time data servers, session managers for collaborative VEs, etc.) using a variety of communication protocols. Note that while Figure 1 shows only a single client per application, in fact the same application may run on many cooperating and interacting clients.

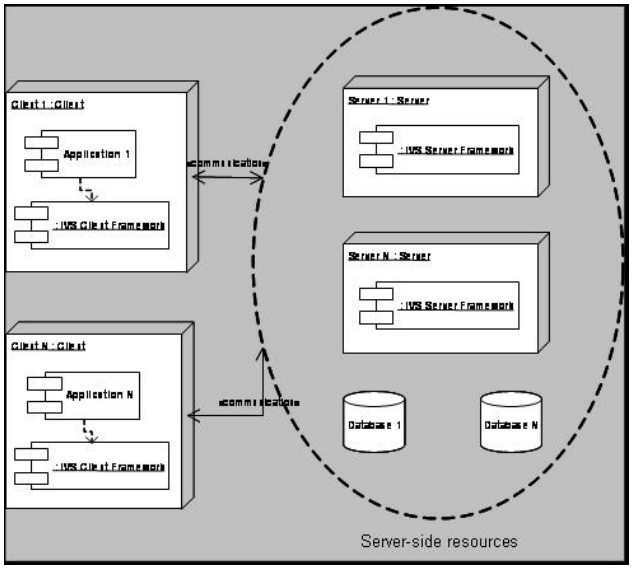

Figure 1. High-level view of the IVS framework

It is reasonable to ask whether a special framework for training and operations is necessary, especially in light of the many toolkits that already exist (such as OpenGL [8-10] and DirectX [11] for 3D graphics, and the Java API [12] and the .NET Framework Class Library [13-15] for general software development). First-hand experience with a variety of different NASA teams suggests there is in fact, a strong need for such a framework, one that builds upon the aforementioned toolkits. Its existence would not only result in significant cost savings for NASA, but also promote knowledge capture through the publication of design patterns common to spaceflight training and operations.

Experience and iteration have suggested a grouping of components in the framework as shown in Figure 2:

1) *Information Management*
2) *Visualization*
3) *Simulation and Decision Making*
4) *Real-time Data*.

Attention is restricted to the client-side components as they have been the focus of the work accomplished to date. The client-side components of an application would deploy some or all of these components.

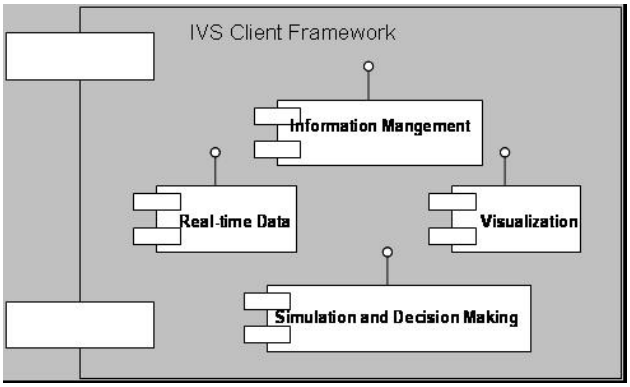

Figure 2. Client-side subcomponents

Information Management refers to accessing and potentially modifying document-type data: design review documents, images, problem reports, procedure specifications, test results, etc. *Visualization* refers to VEs of spacecraft and GUI widgets useful for presenting data. *Simulation and Decision Making* includes components designed to do trade-off analyses, procedure generation, and fault detection and identification. *Real-time Data* includes real-time information associated with the spacecraft, voice and video data, and security mechanisms to guarantee confidentiality.

In describing the IVS framework, it is equally important to point out what it is not—a one-stop-shopping, complete solution to training and operations software development. It is anticipated that factoring out common code and design patterns will hopefully generate a framework that is useful in many, but not all, settings.

3. Related Work

The notion of looking for design patterns in application areas is well established; [1-3] provide good introductions to the topic. As for successful software frameworks, there are many. Graphics [8-11] and general software frameworks [12-15] have already been mentioned. Specific to training, Tu et al. [16] have looked at the problem of developing a framework to handle audio and video in a collaborative setting, while Anido et al. [17] have investigated the use of CORBA to allow components to communicate in a training framework. Some early work on data-driven interactive 3D graphics applications was done by Ashbaugh et al. [27].

Virtual environments can be a critical component of training and operations software. Representative existing toolkits include VESS [18], MASSIVE [19, 20] and NPSNET [21, 22]. A recent survey of VEs can be found in [23]. Kapolka et al. [4] have investigated using component frameworks to allow dynamic extension of VEs. Security issues in VEs (which all too often are considered late in the design process) have been examined by Salles et al. [24].

4. Framework Details

Of the components shown in Figure 2, the ones that have been most extensively investigated to date are the *Information Management* and *Visualization* components. Work on the *Simulation and Decision Making* component has been done (e.g., procedure generation) but is not presented here; other work has yet to be incorporated into the framework [25]. In addition, *Real-time Data* services are in the early stages of development.

The IVS *Visualization* component provides a reusable way to display a 3D VE of a spacecraft. It is currently implemented in C++ for Windows clients and builds under Visual Studio .NET. It is built on top of the OpenGL framework [8-10].

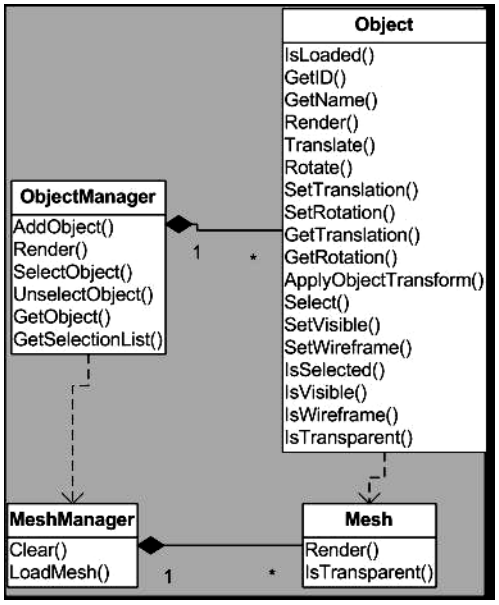

Figure 3. Visualization core classes

A software designer wishing to present clients with a VE would interact with the ObjectManager and Object classes shown in Figure 3 (attributes and non-public operations have been elided from this design-oriented diagram). An application would instruct an ObjectManager to load and render a particular collection of objects, say a set of selected objects in the spacecraft that have real-time telemetry feeds to a ground station. The geometry of an Object is specified in a Mesh. This allows easy geometry reuse in the event that two or more Objects in the scene have identical geometry.

The current framework makes geometry available for components of the ISS. Certain modules have higher levels-of-detail than others. All geometry is contained

within the Visualization component for use client-side by the application.

A non-trivial issue being addressed in the framework is the process of repurposing CAD to generate the set of graphical models for real-time rendering. With a multitude of engineers and contractors designing and building the ISS, each designing their segments with different CAD packages, the framework is confronted with importing CAD from different vendors into the VE coherently. Additionally, using the CAD geometry without simplification is ill suited for interactive real-time rendering. Displaying and rendering very high level-of-detail views of a spacecraft such as the ISS is unachievable even with state of the art graphics hardware, let alone those possessed by even the most advanced laptop computers. The process of repurposing CAD includes off-line automated polygon reduction techniques, converting CAD files into an optimized file-format for interactive graphical rendering. Current work is looking to increase the automation of this pipeline still further, reducing the need for manual intervention.

The design shown in Figure 3 does not intrinsically provide an application designer with a technique to handle varying scene complexity. This is pertinent to complex spacecraft, and the tradeoff is one of scene complexity versus rendering frame rate on the client machine. To have the broadest possible appeal, the IVS *Visualization* component will look to provide level-of-detail support in future iterations.

The *Information Management* component shown in Figure 2 is intended to provide the capability to retrieve and modify document-type data (this component has been previously discussed and thus will only be summarized here; see [26] for further details). Any space program has associated with it an overwhelming number of documents. This includes design review documents, images, procedure specifications, test data and the like. It would be impractical to require all types of data to reside in a single location—for the ISS, there are many international partners and a wide range of types of data. ISS onboard inventory information is maintained in one database (the Inventory Management System), design documents in other repositories (e.g., the Vehicle Master Database), problem reports in yet other sources (e.g., the Problem Reporting and Corrective Action database), and so on. Developers needing access to this information would benefit from client-side components that could easily query these various sources. The approach taken to date in the IVS *Information Management* component has been to copy documents from ISS data sources, implement a new schema, and then provide a simple query facility based on a part identifier.

5. Framework Application: SSTF

Having described the general features of the IVS framework, a specific example of its use is now presented. The Space Station Training Facility (SSTF), located at the NASA Johnson Space Center in Houston, Texas, is one of the principal astronaut training facilities associated with the ISS. It houses physical mock-ups of ISS modules that have a high degree of functional accuracy. The mocked-up modules have items like caution and warning panels, audio terminal units, and a variety of other systems the astronauts actually use when onboard to interface with the station. All these systems are connected to a hardware and software back-end at SSTF that allows real-time simulation of station conditions. From a functional point of view, the only facility more accurate than SSTF is the actual ISS.

SSTF had an interest in looking at alternative interfaces to some of their training systems, to give astronauts the ability to engage in remote training. Users at remote locations (for example at Star City, Russia) would interact with a portable or desktop computer to navigate through a VE of the ISS. Certain objects in the VE would be linked to their underlying simulations at SSTF. As well, users would be able to retrieve supporting documents describing the training procedure they were involved in.

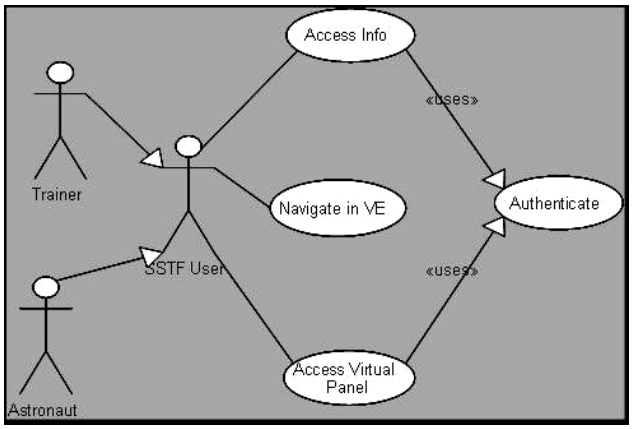

Figure 4. SSTF uses cases

These requirements could be distilled to three principal use cases as shown in Figure 4: accessing information, navigating in a VE of the ISS, and accessing virtual panels. Two principal classes of users exist: astronauts engaged in training and SSTF trainers overseeing the training process.

Virtual panels (VPs) require some elaboration. These are 2D GUI widgets that, when properly connected to server-side resources at SSTF, provide an interface to (and can fully control) a Station component in the

simulation. As an example, a caution and warning panel physically located in the mock-up would have a corresponding VP. If a caution were triggered in training, the caution light would be visible both in the physical mock-up and on any computer at SSTF that had the corresponding VP displayed. The caution could be acknowledged by physically pressing a button in the mock-up or by pressing a button on the VP. VPs are used by instructors to monitor training and run sample scenarios from their Instructor Operator Stations. They are also used to provide a virtual interface to components that do not yet physically exist at SSTF.

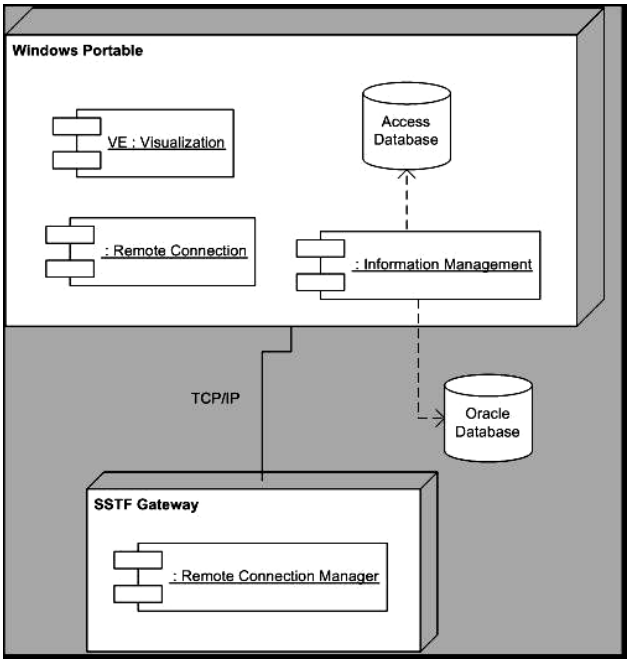

Figure 5. SSTF deployment diagram

Components from the IVS framework were used as shown in Figure 5 to meet the SSTF requirements. Some components had to be uniquely developed to interface with existing SSTF simulation programs. The client-side components were written in C++ and are being deployed on portable computers running Windows 2000/XP (a typical client having a Pentium 4 1.7 GHz CPU with a GeForce 4 440 Go graphics card). A client machine in turn communicates with a gateway machine at SSTF, which handles requests for specific virtual panels. The server-side Remote Connection Manager is written in Perl (version 5.x). Authentication is handled via a combination of passwords and physical security devices. Not shown in Figure 5 are the various interactions between the gateway machine and the extensive network of simulation computers at SSTF. Document-type data describing a particular training scenario can be accessed either from a Microsoft Access database stored locally on the client or from a remote Oracle database.

Figures 6 and 7 show screen shots of the SSTF application. Figure 6 shows a user navigating to obtain an exterior view of the ISS (as it will nominally appear when station assembly is complete). Figure 7 shows a user about to interact with a virtual panel of a Portable Computer System (PCS). PCSs are computers that astronauts actually use when onboard the ISS to monitor and control systems. The virtual PCS has a matching user interface, when used, instead of controlled the actual ISS, it controls the simulation at SSTF.

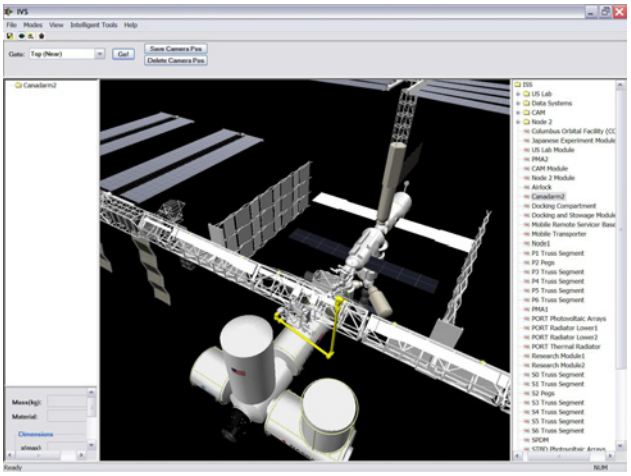

Figure 6. A screen shot of the SSTF application

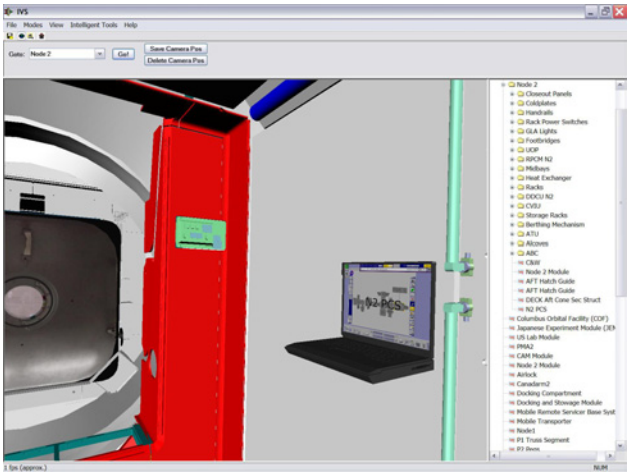

Figure 7. A screen shot of the SSTF application showing a Portable Computer System in Node 2

The net effect of linking the VPs to the virtual environment is that users can engage in remote virtual training from secure locations. A sense of spatial awareness of components is maintained, and the complete functionality offered by the VPs is available.

The application under development for SSTF continues to be refined. Improvements are being made to some of the server-side components. As well, geometry is being added to the VE to accurately model the various virtual panels available at SSTF. Lessons learned from its development are being fed back into revising and updating the IVS framework. The application is currently being used at SSTF for demonstration purposes only; actual deployment will follow only if the application meets or exceeds the high standards SSTF demands of training tools.

6. Conclusions and Future Work

Although the IVS framework will continue to mature, it has already demonstrated its ability to speed application development in different training and operations scenarios through component reuse. That said, the IVS framework is still preliminary in nature and likely needs an additional eighteen months of development effort before it can usefully be made available to the larger spaceflight community. It seems almost certain that frameworks such as the IVS will become more common within the spaceflight software community within the next decade.

Another conclusion, and a major lesson drawn from this work, is the absolute necessity of working early and often with a variety of teams involved in training and operations. Their involvement is critical in defining and understanding the important details of each application area, which in turn increases the likelihood that the framework will be effective and useful.

Future work will look to expand the *Real-time Data* and *Simulation and Decision Making* components of the framework, providing real-time data from spacecraft components as well as securely handling voice and video traffic during remote training. The VE components will be expanded to include a collaborative component, building on the work of NPSNET and others [18-22]. The usefulness of including ambient sound in the VE for purposes of training will be investigated (the ISS, for example, can be a relatively loud environment in which to work). Current and future work on the framework is increasingly building on Microsoft's .NET framework [13-15]. Finally, the *Visualization* component will look to provide more detailed and varied geometry for the ISS and other spacecraft, level-of-detail services appropriate for training and operations, and assembly sequence support for the ISS.

7. Acknowledgements

The authors gratefully acknowledge the help and support provided by Michael Belansky and Louis Malone, II of the Space Station Training Facility at the NASA Johnson Space Center.

Work was funded by the NASA Computing, Information and Communications Technology Program under the Computing, Networking and Information Systems Project.

8. References

[1] E. Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison-Wesley, Reading, MA, 1995.

[2] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process,* 2nd ed., Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[3] A. Shalloway and J.R. Trott, *Design Patterns Explained: A New Perspective on Object-Oriented Design,* Addison-Wesley, Boston, MA, 2002.

[4] A. Kapolka, D. McGregor, and M. Capps, "A Unified Component Framework for Dynamically Extensible Virtual Environments," *Proc. 4th Int'l Conf. Coll. Virt. Env.* (CVE'02), Bonn, Germany, Sep. 2002, pp. 64-71.

[5] NASA Human Spaceflight, International Space Station, http://spaceflight.nasa.gov/station/ (verified Mar. 2003).

[6] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual,* Addison-Wesley, Reading, MA, 1999.

[7] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide,* Addison-Wesley, Reading, MA, 1999.

[8] OpenGL Home Page, http://www.opengl.org (verified Mar. 2003).

[9] M. Woo et al., *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2,* 3rd ed., Addison-Wesley, Reading, MA, 1999.

[10] D. Shreiner et al., *OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.2,* 3rd ed., Addison-Wesley, Reading, MA, 2000.

[11] DirectX Home Page, Microsoft Corporation, http://www.microsoft.com/windows/directx/ (verified Mar. 2003).

[12] Java API Documentation, Sun Microsystems, http://java.sun.com/apis.html (verified Mar. 2003).

[13] .NET Home Page, Microsoft Corporation, http://msdn.microsoft.com/netframework/ (verified Mar. 2003).

[14] .NET Framework Class Library, Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/cpref_start.asp (verified Mar. 2003).

[15] J. Richter, *Applied Microsoft .NET Framework Programming,* Microsoft Press, Redmond, WA, 2002.

[16] S. Tu, L. Xu, and Y. Wu, "A Framework Approach to Accommodation of Multimedia Communication for Training Systems," *Proc. of the IEEE 4th Int'l Sym. Multimedia Software Eng.* (MSE'02), Newport Beach, CA, Dec. 2002, pp. 232-239.

[17] L. Anido et al., "Architecting CORBA-based Frameworks to Support Distributed and Interoperable Training Systems in Large Enterprises," *Proc. 11th Int'l Workshops on Enabling Tech.: Infra. for Collaborative Enterprises* (WETICE'02), 2002, pp. 170-175.

[18] J. Daly, B. Kline, and G.A. Martin, "VESS: Coordinating Graphics, Audio, and User Interaction in Virtual Reality Applications," *Proc. Virt. Reality 2002* (VR 2002), Orlando, FL, Mar. 2002, pp. 289-290.

[19] J. Purbrick and C. Greenhalgh, "An Extensible Event-based Infrastructure for Networked Virtual Worlds," *Proc. IEEE Virt. Reality 2002* (VR'02), Orlando, FL, Mar. 2002, pp. 15-21.

[20] C. Greenhalgh and J. Purbrick, "Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring," *Proc. 3rd Int'l Conf. Coll. Virt. Env.* (CVE 2000), San Francisco, CA, Sep. 2000, pp. 119-127.

[21] M. Capps et al., "NPSNET-V: A New Beginning for Dynamically Extensible Virtual Environments," *IEEE Computer Graphics and Applications,* vol. 20, no. 5, Sep/Oct. 2000, pp. 12-15.

[22] NPSNET Home Page, NPSNET Research Group, http://www.nspnet.org/~npsnet/ (verified Mar. 2003).

[23] K. M. Stanney (Ed.), *Handbook of Virtual Environments,* Lawrence Erlbaum Assoc., Mahway, NJ, 2002.

[24] E.J. Sallés et al., "Security of Runtime Extensible Virtual Environments," *Proc. 4th Int'l Conf. Coll. Virt. Env.* (CVE'02), Bonn, Germany, Sep. 2002, pp. 97-104.

[25] E. Wilson, C. Lages, and R. Mah, "Gyro-based Maximum-likelihood Thruster Fault Detection and Identification," *Proc. 2002 American Control Conf.,* Anchorage, AK, May 2002, pp. 4525-4530.

[26] B.J. Betts et al., "A Data Management System for International Space Station Simulation Tools," *Proc. 2002 Int'l Conf. App. Modeling and Simulation* (AMS 2002), ACTA Press, Boston, MA, Nov. 2002, pp. 500-504.

[27] J.B. Ashbaugh, D.P. Roland, and L.F. Laird, "DSPOBJ—System for Display of Multiple Sets of Three-Dimensional Data," *Comput. & Graphics,* vol. 3, 1978, pp. 63-70.